US 20190104059A1

(19) **United States**

(12) **Patent Application Publication**

Surcouf et al.

(10) **Pub. No.: US 2019/0104059 A1**

(43) **Pub. Date: Apr. 4, 2019**

(54) **TEMPLATE-COMPATIBLE ENCODING FOR CONTENT CHUNK AGGREGATION AND MAPPING**

(71) Applicant: **Cisco Technology, Inc.**, San Jose, CA (US)

(72) Inventors: **Andre Jean-Marie Surcouf**, St Leu La Foret (FR); **William Mark Townsley**, Paris (FR)

(21) Appl. No.: **16/094,963**

(22) PCT Filed: **Apr. 17, 2017**

(86) PCT No.: **PCT/US2017/027996**

§ 371 (c)(1),
(2) Date: **Oct. 19, 2018**

Related U.S. Application Data

(60) Provisional application No. 62/324,696, filed on Apr. 19, 2016, provisional application No. 62/324,710, filed on Apr. 19, 2016, provisional application No. 62/324,657, filed on Apr. 19, 2016, provisional application No. 62/324,727, filed on Apr. 19, 2016, provisional application No. 62/324,721, filed on Apr. 19, 2016, provisional application No. 62/340,156, filed on May 23, 2016, provisional application No. 62/340,182, filed on May 23, 2016, provisional application No. 62/340,162, filed on May 23, 2016, provisional application No. 62/340,174, filed on May 23, 2016, provisional application No. 62/340,166, filed on May 23, 2016.

(30) **Foreign Application Priority Data**

Jan. 11, 2017 (GB) 1700470.6

Publication Classification

(51) **Int. Cl.**

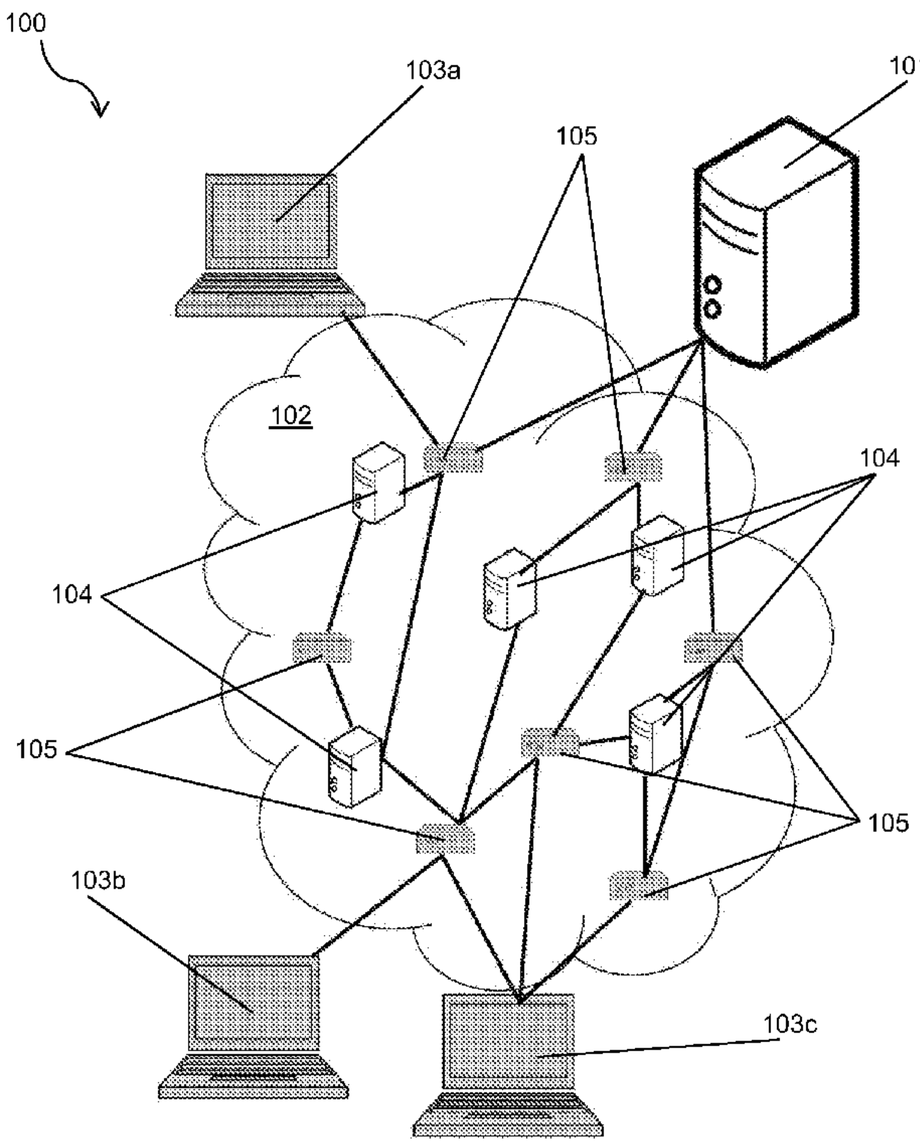
H04L 12/721	(2006.01)
H04L 12/749	(2006.01)
H04L 29/12	(2006.01)
H04N 21/643	(2006.01)
H04N 21/845	(2006.01)
H04N 21/658	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 45/34*** (2013.01); ***H04L 45/741*** (2013.01); ***H04L 61/1511*** (2013.01); ***H04L 61/6059*** (2013.01); ***H04N 21/6581*** (2013.01); ***H04L 61/2007*** (2013.01); ***H04N 21/64322*** (2013.01); ***H04N 21/8456*** (2013.01); ***H04L 61/251*** (2013.01)

(57) **ABSTRACT**

A method of enabling access to content in a network implementing Internet Protocol version 6 (IPv6) is described, the method including accessing a content addressing file including entries each comprising a content portion location associated with the content portion. The content portion location associated with the content portion is extracted for an entry and, based on the content portion location, a section of an IPv6 address for the content portion is formed. Methods of addressing content for storage and retrieving content are also described.



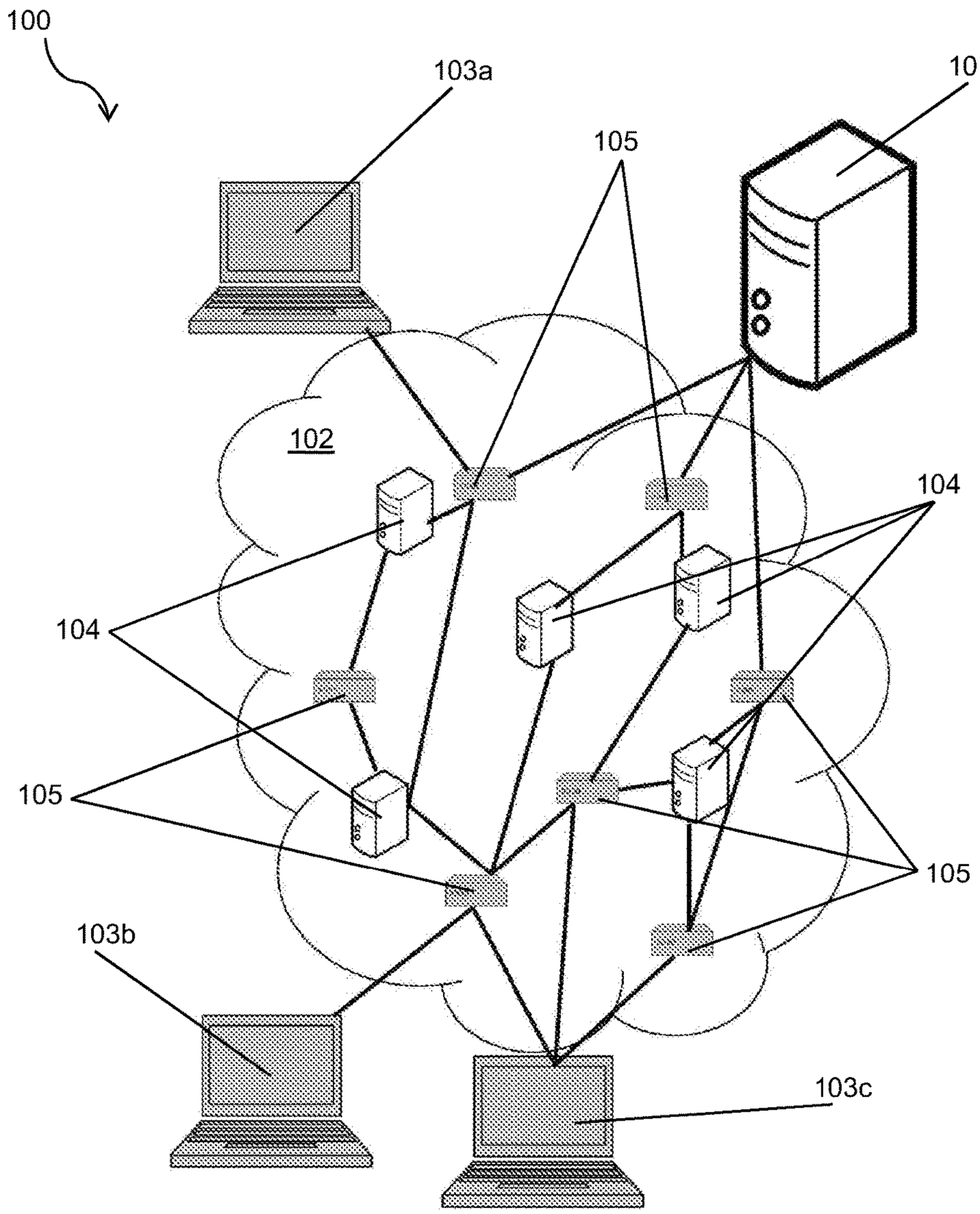


Fig. 1

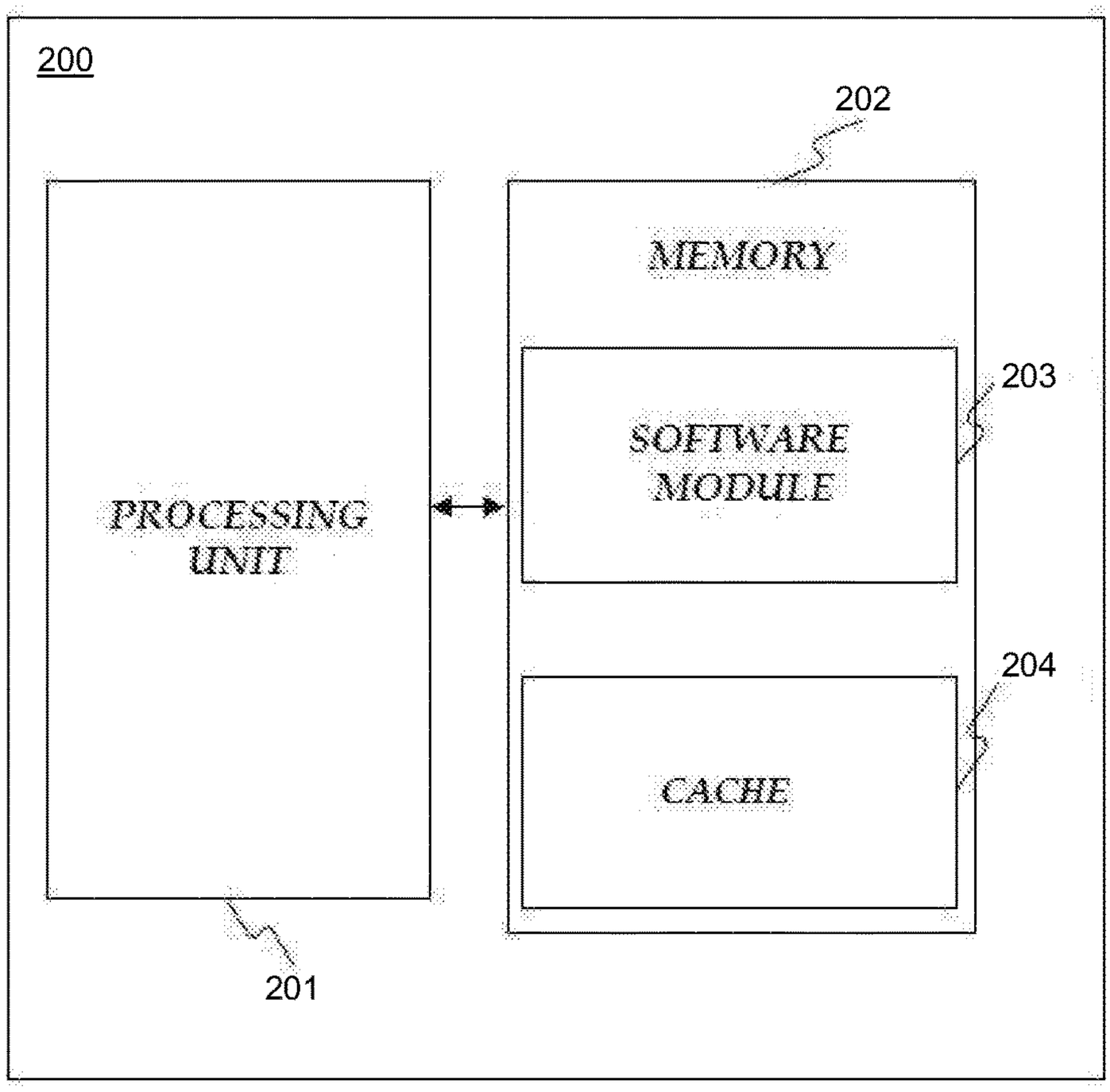

200
202
MEMORY
203
SOFTWARE MODULE
PROCESSING UNIT
204
CACHE
201

Fig. 2

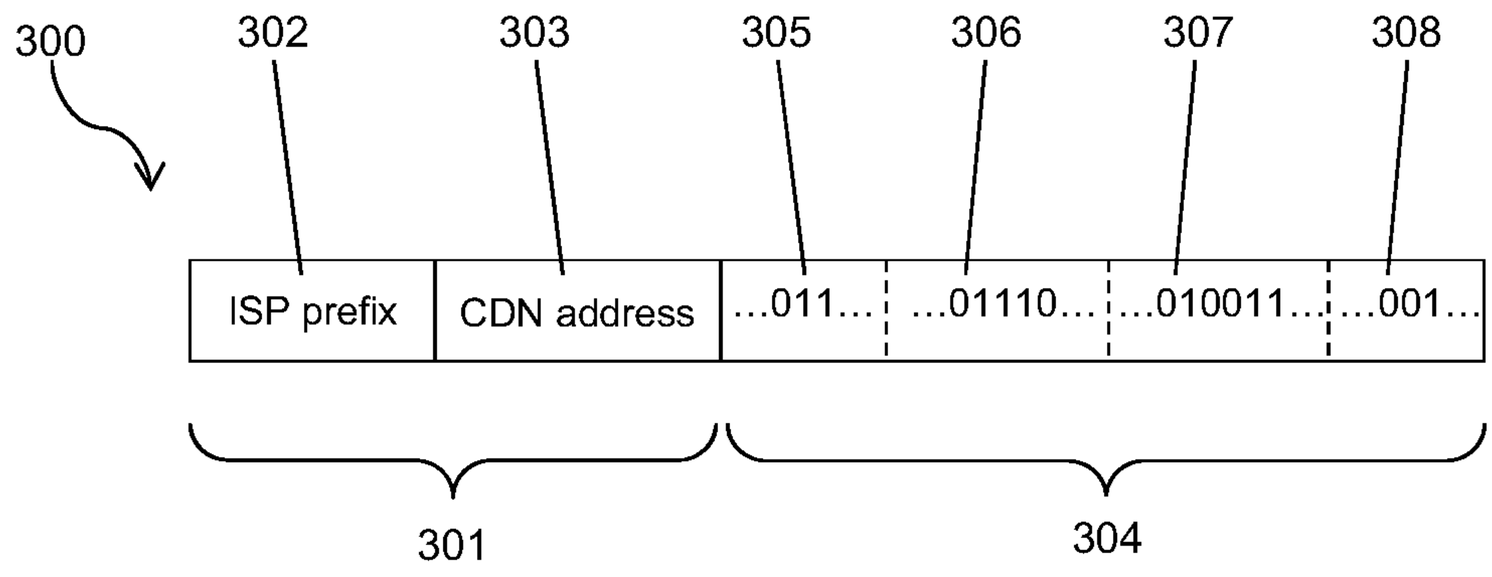

300
302
303
305
306
307
308
ISP prefix
CDN address
…011…
…01110…
…010011…
…001…
301
304

Fig. 3

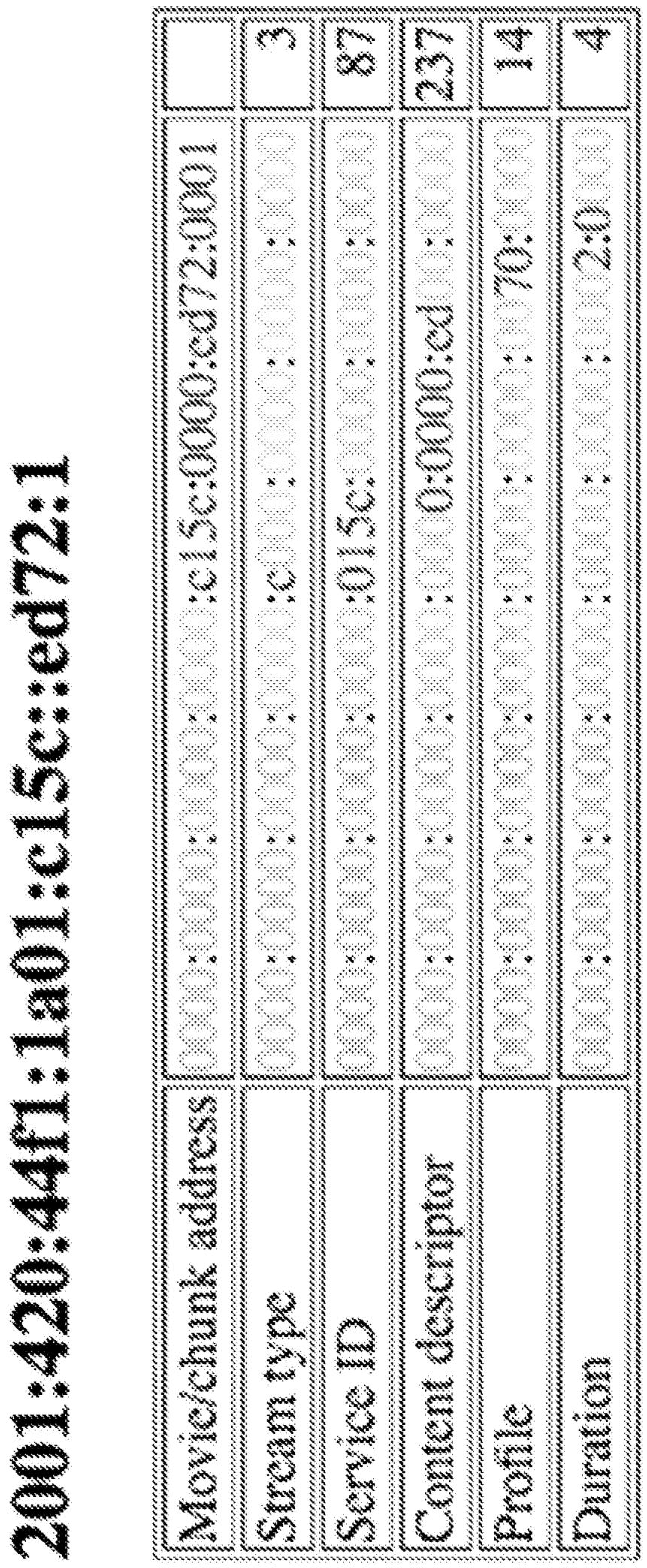

2001:420:44f1:1a01:c15c::ed72:1
Movie/chunk address
Stream type
Service ID
Content descriptor
Profile
Duration
2001:420:44f1:1a01:c15c::ed72:1
2001:420:44f1:1a01:c15c:0MM0:ed72:1
0:0:0:0:0:0:0:0
0:0:0:0:0:0:0:0
0:0:0:0:0:0:0:0
0:0:0:0:0:0:0:0
0:0:0:0:0:0:0:0
0:0:0:0:0:0:0:0
4
14
237
87
3

Fig. 4

TEMPLATE-COMPATIBLE ENCODING FOR CONTENT CHUNK AGGREGATION AND MAPPING

TECHNICAL FIELD

[0001] The present disclosure relates generally to content retrieval in a network, in particular in relation to issues that arise as networks transition from Internet Protocol (IP) version 4 (IPv4) to IP version 6 (IPv6) networks.

BACKGROUND

[0002] In a traditional IPv4 or IPv6 network, addressing schemes have been established based on the nodes of the network such that individual network nodes have associated local or globally-unique addresses. However, using IPv6, options arise for addressing in the network to be arranged around the information itself. Systems and methods described herein address issues arising from the interaction between these networking models, particularly in the retrieval of data or content from such networks.

BRIEF DESCRIPTION OF THE FIGURES

[0003] Embodiments of the method and apparatus described herein are illustrated in the Figures in which:

[0004] FIG. **1** shows an overview of a network in which embodiments may be implemented;

[0005] FIG. **2** shows a block diagram representative of a router or server;

[0006] FIG. **3** illustrates an exemplary embodiment of an IPv6 address;

[0007] FIG. **4** illustrates schematically the structure of an exemplary IPv6 address.

DETAILED DESCRIPTION

Overview

[0008] Described herein is a method of enabling access to content in a network implementing Internet Protocol version 6 (IPv6), the method including accessing a content addressing file, the content addressing file comprising a plurality of entries, each entry comprising a content portion location associated with the content portion, extracting for an entry the content portion location associated with the content portion and forming based on the content portion location a section of an IPv6 address for the content portion.

[0009] Also described herein is a method of addressing content for storage in a storage node of a network implementing Internet Protocol version 6 (IPv6), the method including receiving content from a content source, obtaining an IPv6 address for the content, converting the IPv6 address for the content into a file name compatible with a file system of the storage node, and storing the content at the storage node using the file name.

[0010] Also described herein is a method of addressing content for storage in a storage node of a network implementing Internet Protocol version 6 (IPv6), the method comprising: receiving content from a content source; associating metadata with the received content; generating an IPv6 address for the content wherein the IPv6 address incorporates an identifier of the content and the metadata; and storing the content associated with the generated IPv6 address in the storage node of the network.

[0011] Also described herein is a method of retrieving content stored in a network, the method comprising: receiving a request for content from a requesting node in an IPv6 network, the request comprising an IPv6 address associated with the content; mapping the received request to a file name for the content based on the IPv6 address in the received request; identifying the content using the file name mapped to the IPv6 address; and transmitting the requested content to the requesting node.

[0012] There are also described herein apparatus, network devices and networks for implementing the methods described herein. In addition, a computer program, computer program product and computer readable medium comprising instructions for implementing the methods described herein is disclosed.

EXAMPLE EMBODIMENTS

[0013] Networks such as local area networks and wide area networks can be implemented between nodes or network devices and are often utilised for distributing data for storage within the network device and for later retrieval of that data. One example of a wide area network is the internet. Nodes of the network may request data from one another. They can do this in one of two ways they can either address a request to another node, the request including details of the data being requested, or they can address a request directly to the required data.

[0014] The network forming the internet is made up of a large number of interconnected nodes. These nodes include clients, switches, servers, routers and other such devices and the nodes can communicate using many different protocols at different layers of the OSI model, but in particular the Internet Protocol version 4 (IPv4) communications protocol. Nodes in the network can be addressed using static or dynamic IPv4 addresses, some of which are globally-reachable but many of which are specific to a local network segment

[0015] FIG. **1** shows a standard network configuration **100** with clients (or consumers, or users) **103***a-c*, a main server **101**, routers **105** and caches **104**. Note that identical numbering has been used for features which are functionally equivalent to one another, e.g. all the caches **104**, and all the routers **105**. This should be interpreted as meaning that each cache has broadly the same functionality as each other cache, although the exact content stored on each, and the technical capabilities of each may vary. Similarly, each router **105** is arranged to operate in broadly the same way as each other router, and importantly they are all interoperable with each other, but specific functioning and capabilities may vary between routers.

[0016] FIG. **2** shows a block diagram of a network node having a network cache **200**. The cache has within a cache memory **202** that is used to store content that can be accessed by other nodes on the network. Addresses in this memory **202** are assigned by several algorithms; most commonly the content most recently requested by one of the end point nodes that the server serves are stored.

[0017] FIG. **1** shows clients **103***a-c* and illustrates how they can receive data over a network **102**, which could be the internet, for example. A request from a client **103** is forwarded to a cache **104**, based on known routing policies. If the cache **104** does not have the exact content requested, it

can either redirect the request to another node, for example it may redirect the request to a main server **101** that is the provider of the content.

[0018] Typically, routing is performed using Internet Protocol (IP) addresses. The IP version currently in use is IPv4, which uses 32 bits to provide a unique address to every node on a network. This provides a little over 4 billion addresses, and it has been recognised for some time that the rate of growth of the internet is quickly rendering this number inadequate. To solve this problem, a new version of the Internet Protocol has been developed. This new version, IPv6, uses 128 bit addresses, allowing a total of around 3.4×1038 addresses.

[0019] A server usually serves a plurality of endpoint nodes across the network as is shown in FIG. **1**. This means that the server may have to deal with multiple requests at the same time. If these requests together ask for more resources than the server or network (e.g. network bandwidth) can provide, then load balancing may be required. Load balancing is where traffic from nodes is redirected and distributed across a group of servers so that the requests can be fulfilled. It may also be the case that the requests cannot be fulfilled. For example many nodes may request a high quality of multimedia content and it may not be possible to deliver this simultaneously to all of the nodes. Therefore an inferior level of content may be delivered to at least some of the nodes in order not to overload the network but nevertheless still transmit the content (albeit at a lower quality than requested) to the nodes.

[0020] IPv6

[0021] IPv6 is an updated version of the internet protocol and is intended to replace IPv4. IPv4 and IPv6 are not designed to be interoperable. IPv6 uses a 128 bit address and these addresses are represented by eight groups of four hexadecimal digits.

[0022] IPv6 networks provide auto-configuration capabilities, enabling automatic assignment of an IP address to a device for example based on the device's Media Access Control (MAC) address. IPv6 networks are simpler, flatter and more manageable, especially for large installations. Direct addressing of nodes from anywhere in the network is possible due to the vast IPv6 address space, which enable the use of globally-unique addresses, and the need for network address translation (NAT) devices is effectively eliminated. An IPv6 address is designed to consist of two parts, a 64 bit prefix that is used for routing and a 64 bit interface identifier. The interface identifier was originally designed to identify a host's network interface, for example by using a hardware address such as the MAC address of the interface. However it has been appreciated that the interface identifier does not have to be used for this purpose. To this end some embodiments described herein utilise the interface identifier to identify content instead. Therefore content, whether it is a page, article or piece of multimedia content, can have its own IP address. This means that instead of routing to a particular node in the network, traffic will route to specific content. Large data files such as streams of media content, are often divided into chunks or segments of data as described below and, in this case, each chunk will preferably have its own address. An IPv6 network implementing this idea of content-addressing may be termed a 6CN or IPv6 Content Network.

[0023] FIG. **3**A illustrates an IPv6 request, showing the structure of the 128 bit address. The address comprises an initial portion known as a prefix, which typically indicates the content provider, and helps to route the request to the correct nodes. The rest of the address narrows down to the specific content required. For example, this may be broadly the title of a movie, which may narrow further to a specific chunk of that movie. Further narrowing may occur when variants of this chunk are specified by progressively later bits of the address.

[0024] An advantage of the IPv6 system is that each variant of each chunk can have a separate address. Moreover, as shown in FIGS. **3**A to **3**C, the variants can have similar addresses to one another, such that the same bits in the address of any chunk can be used to relate to the same variant type. To give a specific example, there may be four different resolutions at which any given chunk of content may supplied. Therefore two bits of the address, for example the 90th and 91st bits may represent the four available resolutions. By using the IPv6 addresses in this way, it is simple for the network to determine not just whether it has the requested contents, but also whether it has similar content, for example, the same chunk at a different resolution.

[0025] In this way, parts of the address act like flags to denote various parameters of the chunk to which they relate. Of course, any logical scheme for denoting variants can be chosen by a content provider, depending on the variants they plan to offer.

[0026] When a network receives a request from a client, it examines the full address (which in IPv6 can be the address of the actual content, not just a network node), and determines whether it is aware of a route towards the address. Examples of an IPv6 address is shown in FIG. **3**. In this figure, the IPv6 address **300** has a routing portion **301** at the beginning and a content portion **304** at the end. The routing portion is used to route requests to the correct node in the network, much in the same way that an IPv4 address is used to deliver content and requests to the correct network location. The routing portion **301** comprises an ISP prefix **302** and a CDN address **303**. It is possible, however, to route directly to content, in which case the routing portion **301** may be dispensed with in favour of an expanded content portion **304**.

[0027] The content portion **304** comprises content and variant flags **305**, **306**, **307**, **308**. These are segregated from one another by dashed lines to indicate that the content provider is free to subdivide the content portion **304** according to their own requirements. The content and variant flags may represent various aspects of the content. For example, portion **305** may represent the name of the content, or some other identifier to determine it. To give a specific example, portion **305** may represent the name of a movie, or the title (or the series and episode number) of an episode of a television show.

[0028] To further narrow down to a specific piece of content, portion **306** may represent a variant of the content, for example, it may represent content having a particular bit rate, or screen resolution, or in a particular file format.

[0029] Narrowing yet further, portion **307** may specify a particular chunk of the content, for example the first 5 minutes of the movie designated by portion **305**, while portion **308** may represent further information defining the chunk. In the event that content has not been chunked, the fields identifying chunk information can be set to 0, as a

default to indicate that there is no chunk information, as the content has not been broken down into chunks.

[0030] Note that by defining a set of rules for interpreting addresses, particular chunk of content, once a piece of content, e.g. a movie has been allocated an portion of the address space, the addresses of the chunks, and the addresses of the variants of the chunks are easy to determine. Of course, the order in which portions **305**, **306**, **307** and **308** are presented can be varied according to a particular content provider's requirements. Indeed, there may be more or fewer subdivisions of the content portion **304**, according to specific requirements.

[0031] Domain Name System

[0032] The DNS is a decentralised naming system for nodes connected to the internet. It is used to associate URLs or domain names with IPv4 addresses. DNS can be implemented to work the same way with IPv6, however now it can also associate content, or a combination of content name and URL with an IPv6 address.

[0033] Information Centric Networking (ICN)

[0034] Information Centric Networking (ICN) provides a network paradigm in which the data or information in the network forms the basis around which information is distributed and routed within the network.

[0035] Each chunk of content has an individual ICN name, which is usually largely human-readable, for example cisco.com/newsitem1.txt/chunk1. An ICN network routes data and requests for that data based on the ICN name. In particular, data chunks are requested by name by ICN applications in content requests or interest packets. Routing techniques, in particular hierarchical routing techniques such as longest-prefix matching, are then applied in order to locate and retrieve the content. A whole item of content is obtained by requesting all of the chunks that make up that content and a list of the relevant chunks can be obtained from a manifest file or data file listing the relevant chunk names. Sometimes an intelligent ICN application can predict multiple chunk names based on a name of a single chunk, for example cisco.com/newsitem1.txt/chunk2 might follow cisco.com/newsitem1.txt/chunk1.

[0036] Reverse path information can be inserted into the request or interest packet on its route to the content so that the network knows how to route the data back though the network to the requesting device. More commonly, however, reverse path information can be stored in the nodes on the outbound path in a technique that can be termed "bread-crumb" routing, so that the content can follow the bread-crumb trail of small data packets left in the outbound network devices, to get back to the requesting network device.

[0037] Chunks of data are commonly cached in intermediate nodes on the way back through the network. These data chunks can be served directly from those caches in order to serve later requests.

[0038] As described in more detail below, ICN applications can control the rate at which requests for data are sent, hence adjusting the sending rate of the content chunks to enable congestion control. In this way, receiver driven transport protocols can be implemented, which are driven by the device requesting and receiving the content. An ICN transport protocol can also further segment the chunks to enable transmission over the network as necessary.

[0039] Internet-layer protocols such as IPv4 and IPv6 are not directly relevant to the implementation of a pure ICN network, but they are sometimes present in the network, in particular as an underlay for example to create point-to-point UDP tunnels.

[0040] Chunking and Retrieval of Media Content in a Network

[0041] Media content (both audio and video) can be divided into chunks or segments for both storage in and delivery over a network. In that way, for example, media content that can be of many hours duration (such as a film or broadcast of a sporting event) can be divided into a number of segments of shorter playback time (such as between 1 second and 1 minute, preferably 1-30 seconds).

[0042] When a network device, such as a client end user device, requests particular media content, such as a particular video file, it needs to obtain all of the chunks of data that make up that media content.

[0043] One way of streaming media content using chunking is to use a technique such as Dynamic Adaptive Streaming over HTTP (DASH) or HTTP Live Streaming (HLS), which allow adaptive bit rate streaming of media content, stored as chunks in a network of one or more HTTP servers, to a network destination requesting the data.

[0044] Prior to storage, the media content is divided into shorter chunks or segments and alternative versions of each chunk are stored at various servers in the network. The alternative versions may be, for example, encoded at different bit rates or may have different formats for playback through a variety of different end user devices (Internet connected TVs, set top boxes, mobile devices including smartphones, laptops etc.)

[0045] When the content chunks are created, a DASH manifest file is also created, which identifies the chunks of data necessary to recreate the whole stream of media content, including details of alternative chunks (for example those that are encoded at different bit rates). If using HLS, the manifest would be an HLS manifest, for example an .m3u or .m3u8 playlist in which chunks of content are identified by a URL in accordance with HLS syntax and semantics.

[0046] Separate DASH manifest files may be created for different formats or encodings of a particular stream of media content, such that a set top box would be working from a different DASH manifest to that used by a smartphone.

[0047] The DASH manifest typically also includes an indication of the location of each of the chunks. However, when dealing with consecutive or aggregate chunks, a manifest template can be used to group multiple chunks that follow a regular format into one description. This can enable easier parsing of the manifest file.

[0048] Based on the manifest, the end user device can retrieve and reconstruct the full media content at the highest bit rate (or from any other representation described in the manifest) currently available to it over the network. In particular, the end user device can obtain subsequent chunks identified in the manifest while decoding and displaying a current chunk.

[0049] The location of each chunk (or sequence of chunks with a particular encoding or at a particular resolution) can be identified by adding suffixes to a base URL, which can then be interpreted by network components such as DNS servers, to direct the request to an appropriate destination at which the requested content is stored.

[0050] Such a naming scheme can be used in both IPv4 and IPv6 networks and is a consequence of the classical HTTP naming scheme where the URL indicates the server to talk to (the FQDN) then a "path" locally interpreted by the said server pointing at the requested content. e.g. http://<fqdn>/<the-internal-server-location-of-the-piece-of-content-I-want-to-get>

[0051] Hence, location can be specified by adding more specific location details to the right-hand-side of a URL. Hence, in a DASH manifest, once a base URL has been defined (for example a particular server on which a group of chunks is hosted), locations such as directories on a particular server can be further defined by adding location details to the right-hand-side of the base URL. A base URL may be a Fully Qualified Domain Name (FQDN) or absolute address identifying the full location of the relevant server such that it is reachable from anywhere on the network. The FQDN may also be termed a DNS name.

[0052] Typically an entry in the DASH manifest is used to build up a filename Most of the time the model to build a URL used by the device to request a chunk is defined by some XML regular expression called the segment template in DASH formatting, such as:

[0053] <SegmentTemplate timescale="96"media="bunny_$Bandwidth bps/BigBuckBunny_4s$Number.m4s" startNumber="1" duration="384"initialization="bunny_$Bandwidth$bps/BigBuckBunny_4s_init.mp4"\>

[0054] The "Segment Template" XML section also contains information to find the player initialization file (one per bitrate in this example). In addition the manifest also contains other information, or metadata, used by the player to know what the frame rate is etc.

[0055] This template is used by the device to create the right hand side of the full URL, the missing part of the URL (inc the FQDN) conventionally points to the place where the dash manifest is in the HTTP server

[0056] Template Generation

[0057] As described above, chunk locations can be identified in a DASH manifest implemented on an IPv4 network by further specifying a base URL. However, in an IPv6 network in which chunks of data are addressed directly with their own IPv6 addresses, the IPv6 address of each chunk (and potentially each version of each chunk) will be different, so the IPv6 address in the DASH manifest needs to be adapted for each chunk.

[0058] As described in more detail below, in the present IPv6 content-centric network the FQDN (an IPv6 address or a DNS AAAA record) is used as a unique identifier pointing at the requested content.

[0059] Most of the time the content chunks of a given duration (e.g. a 4 sec) are grouped in the same directory in which available bit rates are grouped.

[0060] For example:

```
4sec/
4sec/BigBuckBunny_4s_simple_2014_05_09.txt
4sec/bunny_1008699bps
4sec/ bunny_1008699bps/BigBuckBunny_4s1.m4s
.....
4sec/ bunny_1008699bps/BigBuckBunny_4s10.m4s
```

[0061] In addition to the above, the dash manifest also contains an AdaptationSet giving information such as image resolution, aspect ratio, frame rate etc. An example format might be:

[0062] <AdaptationSet segmentAlignment="true" group="1" maxWidth="480" maxHeight="360" maxFrameRate="24" par="4:3">

[0063] Since 6CN does not require any full URL description, the idea here is to use the same strategy to build up an IPv6 address or an FQDN.

[0064] In an IPv6 content network (6cn), the v6 addresses identifying a chunk follow some coding convention and might look like:

[0065] 6cn_prefix:content_id:profile:duration:chunk_id

[0066] 6cn_prefix identifies a 6cn system without carrying any particular signification in the context of this invention. The skilled persons will appreciate that other address coding conventions could be used within the context of the present system.

[0067] content_id: defines the full content (Big-Buck-Bunny from the above example

[0068] profile: identifies other content attributes such as the codec, the frame rate, the resolution etc.

[0069] duration: chunk individual duration

[0070] chunk_id: current chunk number

[0071] By way of a non-limiting convention, chunk #000 identifies the initialisation file used by the player to initialize the video decoder with parameters applicable to all chunks.

[0072] By way of a further non-limiting convention, the manifest itself is often identified as the last possible chunk number (for example 0xefff if this corresponds to the number of chunks of the content).

[0073] In one implementation, an existing IPv4 manifest may be converted into an IPv6 manifest. Each entry in the IPv6 manifest, which may simply be an IPv6 address, encodes all of the information about the chunk that was previously in the IPv4 manifest.

[0074] The ability can be maintained to translate between a content-based addressing scheme and a server or node-based addressing scheme by reformatting the data to enable back and forth representations between an address identifying content chunks and traditional manifest representations. For example, a mapping may be enabled between a classical URL and a URL directed to the host part when IPv6 addresses are used to directly denote contents. In a classical URL format the host section denoting the server (or its DNS name) can be the IPv6 address of the server.

[0075] To implement this system, it can be convenient to extract the full description from an existing content, in particular when the manifest contains a template used by the player to create each URL chunk. As noted above, the full description cannot be limited to the DASH manifest since the manifest typically only applies to the directory in which it is.

[0076] In the general case, since a DASH manifest is fundamentally a list of URLs the URL chunks could be anything. In particular, a URL contained in the manifest may be a full URL that is directly usable by a server.

[0077] The content_id field of the 6CN address is conventional as it refers to the whole content.

[0078] In a particular exemplary implementation, the full description information is used to code the Sprofile (bunny_1008699 bps combined with internal manifest information

such as the one provided in the AdaptationSet. The $duration field of the 6CN chunk addresses is also extracted from this full description.

[0079] In one embodiment, each .m4s file from the existing hierarchy represents one chunk of a given profile. The file name can therefore be used to create the Schunk_id of the resulting 6CN address.

[0080] For example:

```
4sec/
4sec/BigBuckBunny_4s_simple_2014_05_09.txt
4sec/bunny_1008699bps
4sec/ bunny_1008699bps/BigBuckBunny_4s1.m4s
.....
4sec/ bunny_1008699bps/BigBuckBunny_4s10.m4s
```

[0081] BigBuckBunny_4s10.m4s will get the following 6CN v6 address:

[0082] 6CN_prefix:BigBuckBunny:1280_720_1008699 bps_24 fps_mp4:4 sec:0x0002

[0083] The skilled person will appreciate that this principle can be extended to other content having other associated content names and metadata.

[0084] It also important to note that the above is just an example of how a v6 address can be formed but any other addressing template could work too (e.g. the profile could be encoded as part of content_id, thus a movie in HD and the same in SD would have different content_id.

[0085] In a further embodiment that may be implemented in conjunction with the first implementation, in keeping a mapping that can be addressed using the Template language, we can easily move between a filename, an IPv6 address, and an FQDN (DNS name).

[0086] In particular, in one embodiment, an existing DASH or HLS manifest containing a classical Uniform Resource Locator (URL) such as:

[0087] http://<content-server-fqdn>\<the-server-location-of-the-piece-of-content-I-want-to-get> can be rewritten as a URL with an IPv6 address, the IPv6 address preferably pointing directly to the content in a content networking (6CN) setup such as:

[0088] http://<ipv6-address-of-the-piece-of-content-I-want-to-get>

[0089] The rewritten URL gives access to the same content as the classical URL in a transparent way (from a user device perspective).

[0090] The rewritten URL can be stored in a IPv6-based manifest for the content.

[0091] As will be appreciated by the skilled person from the present disclosure, the rewritten IPv6 address may be generated based on one or more properties of the content including: an identifier of the content provider or source of the content, an identifier of the specific content itself, the server location of the content, a version type, encoding type or length of the content, stream type, service ID, chunk ID, codec, frame rate, content descriptor, profile and duration of the content.

[0092] The section below describes a mechanism for a possible reverse mapping, which can be used internally by the physical streamers to find the actual content corresponding to the v6 address the said streamer received a request for.

[0093] In particular, an ASCII representation of the address can be used to identify the physical file in which the chunk is stored. This just requires renaming the initial .m4s file (or content of another format) with the main difference that this time the new filename gives all the chunk information as it is just another representation (ASCII) of the content information.

[0094] An example might be:

[0095] 6CN_prefix-BigBuckBunny-1280_720_1008699 bps_24 fps_mp4-4sec-0x0002

[0096] The corresponding address v6, 6CN address might be represented as e.g. 2001:420:44f1:1a01:c15c::ed0a:2

[0097] The ASCII representation of the address (which might be used as a local file name) could be something like:

[0098] 2001-420-44f1-1a01-c15c--ed0a-2

[0099] Hence the filename can be used to give all of the chunk information. In particular, when a request is received on the basis of a particular IPv6 address, this is translated to ASCII, which provides the filename to look for on the server, that is it identifies the physical file.

[0100] Similarly the ASCII representation of the 6CN address can be used as an FQDN so that http://6CN_prefix-BigBuckBunny-1280_720_1008699 bps_24 fps_mp4-4 sec:0x0002.6cn.io/is a perfectly valid URL, a DNS lookup giving back the same 6CN v6 address for that chunk.

[0101] All of the above can be applied to any file retrieval system, not just chunked content. In that case, the "chunk ID" field becomes meaningless, but this could be set to a default.

[0102] FIG. **4** illustrates schematically the structure of an exemplary IPv6 address including an element that identifies the chunk address, in this case the second half of the IPv6 address. As illustrated in FIG. **4**, the chunk address can contain information or metadata relating to the chunk of content. In the exemplary embodiment shown, the chunk address is contains information such as stream type, service ID, content descriptor, profile and duration of the content. It will be appreciated that this information can then be extracted from the IPv6 address as the content is requested or delivered through the network, as described herein.

[0103] In a further embodiment, which may be implemented in conjunction with the systems and methods described above, content descriptors and identifiers may be used in order to define a directory structure for the content within a storage node. In particular, in a classical URL such as http://<content-server-fqdn>/<the-server-location-of-the-piece-of-content-I-want-to-get> the <the-server-location-of-the-piece-of-content-I-want-to-get> part can be used to map to an internal file in the cache file system hierarchy. One way to achieve this is to remap the internal file system hierarchy on the URL hierarchy:

[0104] http://content-server/video/4sec/bunny_1008699 bps/BigBuckBunny_4s1.m

[0105] for instance would be internally mapped by the content server to Sdocument_root/video/4 sec/bunny_1008699 bps/BigBuckBunny_4s1.m local file (document_root representing the root of the cache in the local server file system).

[0106] In an IPv6-based 6CN system, however, a URL based on an IPv6 address "http://content_prefix::content_id" could be reformatted and mapped on Sdocument_root/content_prefix/content_id file with "content_prefix" and "content_id" being an ascii representation of the IPv6 address in which for instance all ":" can be replaced by "_" to make the file name compatible with the file system naming convention. Hence the IPv6 address may be used to define a suitable directory structure for storage of content that will be under-

standable by the storage node and the nodes can map the storage file name onto an IPv6 address and vice versa.

[0107] In summary, the methods and systems described herein can be used to allow sequenced chunks of the content that are naturally part of an aggregate described in some dash manifest to be encoded in IPv6 addresses, FQDNs and/or filenames used by a filesystem connected to a content streamer, etc. to allow easy and algorithmic mapping between one another at the content level, as well as the ability to access a single item (chunk) within the aggregate within each encoding. Since this mapping is a pure bijection it is easy to move from the v6 address to filename to FQDN representations spaces.

[0108] Since IPv6 addresses can be globally reachable, the network can determine the server from which to provide the content. Hence a base URL no longer needs to be encoded in a manifest. The content can remain globally reachable even if it moves physical server.

[0109] The present disclosure also envisages one or more computer programs, computer program products or logic encoded in computer-readable media for implementing any method claimed or described herein. It will be appreciated that a computer-readable medium, or a machine-readable medium, may include transitory embodiments and/or non-transitory embodiments, e.g., signals or signals embodied in carrier waves. That is, a computer-readable medium may be associated with non-transitory tangible media and transitory propagating signals.

[0110] Throughout the description, references to components or nodes of the network should be construed broadly, and in particular may comprise several subcomponents or modules working in combination to achieve the stated effects. These subcomponents may themselves be implemented in hardware or software. Likewise, many different components may be combined together as one component, for example a single processor may carry out many functions simultaneously. Similarly, any reference to operational steps may comprise hardware, software, or a combination of the two. As already noted, any method described herein, or any part thereof may be carried out by a computer program, or a computer program product.

[0111] References herein to components being connected to one another should be interpreted as meaning either directly connected, or indirectly connected, for example being connected via other components. Indeed, in the case of complex networks, components may be both directly and indirectly connected to one another. Examples of such connection may commonly include, but are not limited to: electronic connections through wires or cables; fibre optic connections; and wireless communication, for example via radio waves, microwaves or infrared.

[0112] In the present disclosure, references to networks should be interpreted broadly. In particular, the internet is often used as an example of a network, but is not limiting. The principles set out herein are applicable to all networks, comprising a collection of processors connected to one another. Connection may be direct, or via switches and routers. The network may further comprise servers and caches, depending on the exact nature of the network. When storage is discussed herein, this may include, without limitation one or more of magnetic, optical, solid state, volatile or non-volatile memory.

[0113] The steps associated with the methods of the present disclosure may vary. Steps may be added, removed, altered, combined, and reordered without departing from the scope of the present disclosure. Indeed, different aspects and embodiments of the disclosure herein may be combined with one another, in any combination and may be implemented in conjunction with one another in a particular network. In particular, individual components, and systems of components may be combined, the tasks of a single component divided between many subcomponents, or equivalent components interchanged without departing from the principles set out herein. Furthermore, features of one aspect may be applied to other aspects of the system.

[0114] Therefore, the present examples are to be considered as illustrative and not restrictive, and the examples are not to be limited to the details given herein, but may be modified within the scope of the appended claims.

1. A method of enabling access to content in a network implementing Internet Protocol version 6 (IPv6), the method comprising:

accessing a content addressing file, the content addressing file comprising a plurality of entries, each entry comprising a content portion location associated with the content portion;

extracting for an entry the content portion location associated with the content portion; and

forming based on the content portion location a section of an IPv6 address for the content portion.

2. (canceled)

3. (canceled)

4. The method according to claim **1** further comprising generating a URL comprising the section of the IPv6 address formed based on the content portion identifier.

5. The method according to claim **1** wherein each entry further comprises at least one of a content portion identifier and metadata associated with the content portion.

6. (canceled)

7. The method of claim **5** wherein forming the section of an IPv6 address is further based on at least one of the content portion identifier and the metadata associated with the content portion.

8. (canceled)

9. (canceled)

10. (canceled)

11. The method of claim **5** wherein the metadata comprises at least one of:

content portion length;

a codec;

frame rate;

resolution; or

duration.

12. (canceled)

13. (canceled)

14. The method according to claim **1** further comprising changing file name of the content on the server to be based on the IPv6 address.

15. (canceled)

16. (canceled)

17. (canceled)

18. (canceled)

19. A method of addressing content for storage in a storage node of a network implementing Internet Protocol version 6 (IPv6), the method comprising:

receiving content from a content source;

obtaining an IPv6 address for the content;

converting the IPv6 address for the content into a file name compatible with a file system of the storage node; and

storing the content at the storage node using the file name.

20. The method of claim **19** wherein converting the IPv6 address comprises extracting a content provider identifier for the content from the IPv6 address.

21. The method of claim **19** wherein converting the IPv6 address comprises extracting a content identifier from the IPv6 address.

22. The method of claim **21** wherein converting the IPv6 address comprises converting at least one of the content provider identifier and the content identifier to an asci representation.

23. The method of claim **21** wherein converting comprises using the content identifier as a sub-directory name within a directory established using the content provider identifier.

24. The method according to claim **19** wherein the method further comprises retrieving the content from the storage node using the file name and generating an IPv6 address for the content based on the file name.

25. The method according to claim **19** wherein obtaining an IPv6 address for the content comprises extracting an IPv6 address for the content from the received content.

26. A method of addressing content for storage in a storage node of a network implementing Internet Protocol version 6 (IPv6), the method comprising:

receiving content from a content source;

associating metadata with the received content;

generating an IPv6 address for the content wherein the IPv6 address incorporates an identifier of the content and the metadata; and

storing the content associated with the generated IPv6 address in the storage node of the network.

27. The method according to claim **26** wherein the content comprises a chunk or portion of content from a content stream.

28. The method according to claim **27** wherein the content stream comprises a stream of live content.

29. The method according to claim **26** wherein the content comprises a chunk or portion of content from a content stream, the method further comprising inserting metadata into the content stream.

30. The method according to claim **29** wherein the metadata comprises at least one of:

content length;

a codec;

frame rate;

resolution; or

duration.

31. The method according to claim **26** wherein the IPv6 address further comprises an identifier of the location of the content, optionally an identifier of the storage node of the network

32. The method according to claim **26** further comprising generating the metadata for associating with the content based on properties of the content received.

33. (canceled)

34. (canceled)

35. (canceled)

36. (canceled)

37. (canceled)

38. (canceled)

39. (canceled)

40. (canceled)

41. (canceled)

42. (canceled)

43. (canceled)

44. (canceled)

45. (canceled)

46. (canceled)

47. (canceled)

48. (canceled)

49. (canceled)

* * * * *